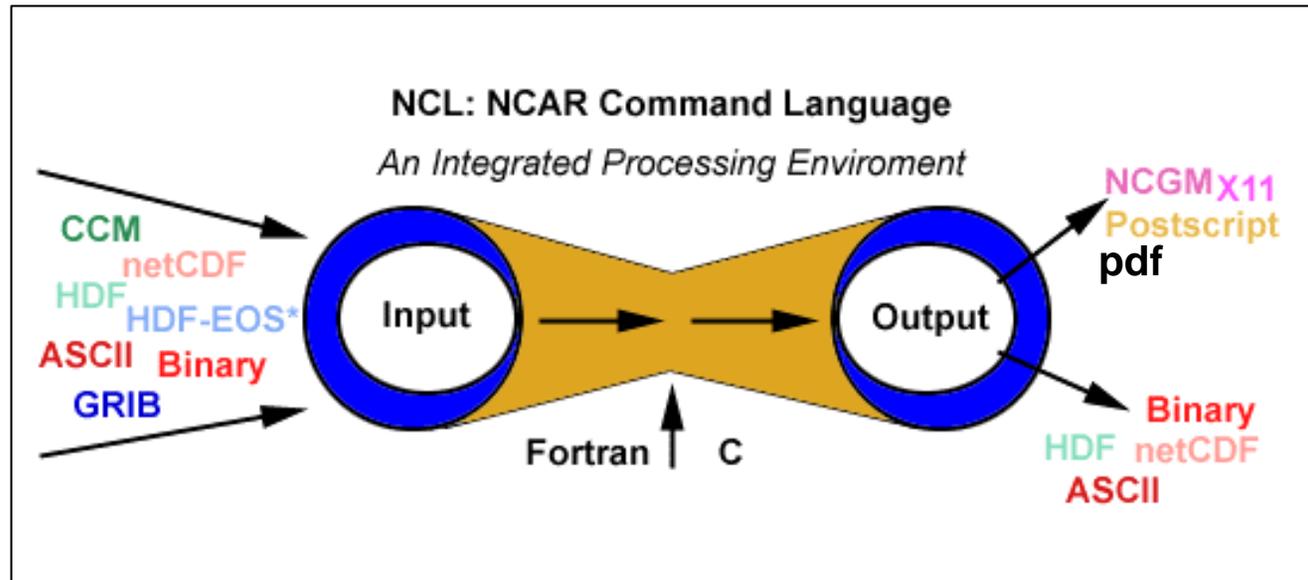


NCL File IO



Dennis Shea

National Center for Atmospheric Research

I/O formats

- **Supported** formats [need not know structure of file]

- ? netCDF [network Common Data Format]

- ? HDF [Hierarchical Data Format (Scientific Data Set only)]

- ? HDF-EOS [Earth Observing System]

- ? GRIB [Grid in Binary; WMO standard; NCEP, ECMWF, FSL]

- ? CCMHT [CCM History Tape; COS blocked only; ccm2nc]

- **Binary**

- ? sequential [F: open(_,form="unformatted", access="sequential")]

- ? flat/direct [F: open(_,form="unformatted",access="direct",recl=_)
[C: write]

- **ASCII**

- ? data organized in columns and rows

- ? use fortran or C to read 'complicated' ascii formats

addfile (1 of 2)

- Used to open a **supported** format only

- **f** = **addfile** (**file_name.ext**, **status**)
 - ? **file_name** => any valid file name; string
 - ? **ext** => extension that identifies the type of file; string
 - ✎ netCDF: "nc" or "cdf" [read/write]
 - ✎ HDF: "hdf", "hdfEOS" [read/write]
 - ✎ GRIB: "grb", "grib" [read only]
 - ✎ CCMHT: "ccm" [read only]
 - ✎ Extension not required to be attached to file
 - ? **status** [read/write status] "r", "c", "w"
 - ? **f**
 - ✎ reference/pointer to file; any valid variable name
 - ✎ may have attributes (**file attributes** or **global attributes**)

<http://ngwww.ucar.edu/ngdoc/ng/ref/ncl/NclFormatSupport.html>

addfile (2 of 2)

- Examples: opening a file**

```
? fin = addfile ("0005-12.nc" , "r")
? fout = addfile (".ncOutput.nc" , "c")
? fio = addfile ("/tmp/shear/sample.hdf" , "w")
? g = addfile ("/dss/dsxxx/Y12345.grb" , "r")
```

- Numerous functions to query contents of supported file**

```
?getfilevarnames
?getfilevardims
?getfilevaratts
?getfilevardimsizes
?getfilevartypes
?isfilevar
?isfilevaratt
?isfilevardim
?isfilevarcoord
```

```
diri = "/fs/cgd/data0/shear/ccm/"
fili = "testCCM"
ext = ".ccm"
fin = addfile(diri+fili+ext , " r ")
```

```
varNames = getfilevarnames (fin)
if (isfilevarcoord(fin, "U", "lat")) then
...
end if
```

Global [File] Attributes

- provide information about the file as a whole (if present)
- examples include:
 - ? convention being used (if any)
 - ? source
 - ? case
 - ? title
 - ? history
 - ? references

```
g      = addfile ("01-11.nc", "r")
atts   = getvaratts (g)      ; get the global attributes
print (atts)                  ; print names of attributes
print (g@title)              ; print just title attribute
```

Import Variable from Supported Fmt

u = f->U

- ? read variable and **all** meta data into memory
- ? no space allowed to left/right of `->` [fatal error]
- ? use **“\$”** syntax to represent variable name if type string

```
f = addfile ("foo.grb", "r")
vNam = getfilevarnames (f) ; all variables on file
      or
vNam = (/ "U", "V" /)      ; manually specify
do n=0,dimsizes(vNam)-1
    x = f->$vNam (n)$      ; $..$ substitute string
    ....
end do
```

u = (/ f->U /)

- read data values **only** and `_FillValue` attribute

printVarSummary(u)

Prints out variable information overview

- ? Type
- ? Dimension information
- ? Coordinate information (if present)
- ? Attributes

Variable: u

Type: double

Total Size: 1179648 bytes
147456 values

Number of Dimensions: 4

Dimensions / Sizes: [time | 1] x [lev | 18] x [lat | 64] x [lon | 128]

Coordinates:

time: [4046..4046]

lev: [4.809 .. 992.5282]

lat: [-87.86379 .. 87.86379]

lon: [0. 0 .. 357.1875]

Number of Attributes: 2

long_name: zonal wind component

units: m/s

Example: open, read, output netCDF

```
begin
;-----
;open file and read in data
;-----
  fin  = addfile ("in.nc", "r")
  u    = fin->U
;-----
;create reference to output file
;-----
  fout = addfile("out.nc", "c")
;-----
;add a global attribute to the file
;-----
  fout@title = "I/O Example 1"
;-----
;Output variable u to the file
;-----
  fout->U = u
end      ; ncl <
```

IO_ex01.ncl

Note: this method of outputting a netCDF file has simple syntax, but can be slow

Example: query file, system commands

```
begin
;-----
; open file, create array of variable names, # of names
;-----
    fin      = addfile (".in.nc", "r")
    vars     = (/ "U", "V", "T" /)
    nvars    = dim sizes (vars)
;-----
; use system to remove output file before creation
;-----
    system("/bin/rm ./out.nc")
    fout     = addfile("./out.nc", "c")
;-----
; loop, query if on the file, then output to netCDF
;-----
    do n=0,nvars-1
        if (isfilevar(fin, vars(n))) then
            fout->$vars(n)$ = fin->$vars(n)$
        end if
    end do
end
; ncl < IO_ex02.ncl
```

Example: new line, output subset

```
begin
-----
; new line [carriage return] is integer representation of 10
-----
;
  nline    = inttochar(10)
-----
;
; open input and output files and read in data
-----
;
  fin      = addfile("in.nc","r")      ; open for read
  fout     = addfile("out.nc","c")     ; open for write
  u        = fin->U
-----
;
; create a global attribute using the new line
-----
;
  fout @title = "This shows how to break " + nline + \
               "a line in two for easier reading" + nline
-----
;
; output a subset of data (from index 3 to 9 by 2)
-----
;
  fout->U_ex03 = u(:,3:9:2,,:,:)
end
; ncl < IO_ex03.ncl
```

Example: only output 3D variables

```
begin
-----
; open file, get list of all variables, and create output file
-----
;
  diri      = "./"                ; current directory
  fili      = "in.nc"
  fin       = addfile (diri+fili, "r") ; open the file
  vars      = getfilevarnames (fin) ; get all variable names on file
  nvars     = dimsizes(vars)      ; number of variables
  fout      = addfile("./out.nc","c")
  fout@source      = "Source Dataset:" + diri+fili
  fout@creation_date = systemfunc("date")
-----
;
; loop through variables. If 3D, output
-----
;
  do n = 0, nvars-1
    ndims = dimsizes(filevardimsizes(fin, vars(n)) ) ; rank [# dim]
    if (ndims.eq.3) then
      fout->$vars(n)$ = fin->$vars(n)$
    end if
  end do
  ; ncl < IO_ex04.ncl
end
```

Import **byte/short** Variable

u = f->U ; read variable and meta data into memory

Variable: u

Type: **short**

Total Size: 294912 bytes
147456 values

byte

147456 bytes
147456 values

[snip]

Number of Attributes: 4

long_name: zonal wind component

units: m/s

scale_factor: 0.15

[slope: 0.15]

add_offset: -3.0

[intercept: -3.0]

(generally) user wants to convert to float

- **COARDS** convention: scale value then add offset

$uf = u * u @ scale_factor + u @ add_offset$

better to use **contributed.ncl [short2flt, byte2flt]**

$u = short2flt(f->u)$; $u = byte2flt(f->u)$

Simple netCDF Creation

```
fout      = addfile (“....nc”, “c”)  
fout@title = "Simple Example"  
fout->U    = u
```

- commonly used

? may be **inefficient** (possibly, **very inefficient**)

? use for file with few variables/records

? can not directly create an unlimited dimension

```
fo      = addfile (“...”, “c”)  
filedimdef (fo, “time”, -1, True )  
fo->U    = u
```

More Efficient netCDF Creation

- **requires explicit definition of file contents**

? must be done in other languages/tools also

- **NCL functions that predefine a netCDF file:**

? **filevardef:** define name of one or more variables

? **filevarattdef:** copy attributes from a variable to one or more file variables

? **filedimdef:** defines dimensions including unlimited dimension

? **fileattdef:** copy attributes from a variable to a file as global attributes

- Less tedious than fortran/C

Example: Create netCDF

```
begin
  T = .....
  fout = addfile("out.nc" , "c")
; create global attributes
  fileAtt = True
  fileAtt@creation_date = systemfunc("date")
  fileattdef (fout, fileAtt)
; predefine coordinate variables
  dimNames = ("/time", "lat", "lon"/)
  dimSizes = (/ -1, nlat, mlon/) ; -1 means unknown
  dimUnlim = (/ True, False, False/)
  filedimdef (fout, dimNames, dimSizes, dimUnlim)
; predefine variable names, type, and dimensions
  filevardef (fout, "time", typeof(time), "time")
  filevardef (fout, "lat" , typeof(lat) , "lat")
  filevardef (fout, "lon" , "float" , "lon")
  filevardef (fout, "T" , typeof(T) , ("/time", "lat", "lon"/) )
; create var attributes (if they do not exist) for each variable
  filevarattdef (fout, "T", T)

; output data values only [use (/... /) to strip meta data]
  fout->time = (/ time/)
  fout->lat = (/ lat /)
  fout->lon = (/ lon /)
  fout->T = (/ T /)
end
```

Writing Scalars to netCDF (hdf)

ncl_scalar is an NCL reserved file dimension name

```
con      = 5
con!0    = "ncl_scalar"
fout     = addfile ("test.nc", "c")
fout->constant = con
```

```
re              = 6.37122e06
re@long_name   = "radius of earth"
re@units       = "m"
fout           = addfile ("test.nc", "c")
filevardef(fout, re, typeof(re), "ncl_scalar")
filevarattdef(fout, "re" , re)
fout->re        = (/ re /)
```

NCL netCDF Creation Process

- **variables**

- ? create variable in memory; assign values
- ? name dimensions **(x!0 = "time")**
- ? assign coordinate variables **(x&time = time)**
- ? define attributes **(time@units = "YYYYMMDD")**

- **create a reference to the file**

- ? **f = addfile** (file_name , "c")
- ? assign global (file) attributes

- **write desired variables**

- simple approach: **f->X = x**
- ? more efficient approach:
 - filevardef, filevarattdef,**
 - filedimdef, fileattdef**
 - f->Y = (/ y /)**

Contents of well written netCDF file_(1 of 2)

- **File attributes that should be included**

- ? title*
- ? source*
- ? Conventions*
- ? history*
- ? institution
- ? creation_date
- ? case
- ? comments
- ? references

```
f@title = "Double CO2"
```

```
f@source = "RCM2 Model"
```

```
f@Conventions = "NCAR-CSM"
```

```
f@history = "ncl < IO_ex01.ncl"
```

```
f@creation_date = systemfunc ("date")
```

```
f@case = "q005.1"
```

```
f@references = "BAMS, 9, July 1999"
```

***NCAR-CSM convention**

Contents of a well written netCDF file (2 of 2)

- **Variables**

- ? **long_name***
- ? **units***
- ? **_FillValue**
- ? **missing_value**
- ? named dimensions
- ? coordinate variable(s)

```
T@long_name = "Temperature"  
T@units = "C"  
T@_FillValue = 1.e+20  
T@missing_value = T@_FillValue  
T!0 = "time"  
T&time=time
```

***NCAR-CSM convention**

Reading Binary/ASCII data

- **7 functions for reading binary:**

- ? **fbincread**: reads multiple unformatted sequential records [Fortran; ieee]
- ? **fbinnumrec**: returns the number of unformatted sequential records [Fortran; ieee]
- ? **fbindirread**: reads specified record from a Fortran direct access file [ieee]
- ? **fbinread**: same as **fbincread** but reads only one ieee rec
- ? **craybincread**: like **fbincread** but for COS blocked data
- ? **craybinnumrec**: like **fbinnumrec** but for COS blocked data
- ? **cbinread**: read binary created via C block IO function "write"

- **1 function for reading ASCII data:**

- ? **asciiread**
- ? use Fortran/C to read anything but simplest ASCII files

- **all above functions allow data to be shaped**

- ? `x = fbincread ("foo_ieee", rnum, (/10,20,30/), "float")`
- ? `a = asciiread ("foo_ascii", (/64,128/), "float")`

Writing Binary/ASCII data

- **4 procedures for writing (ieee) binary data**

- ? **fbinrecwrite**: write unformatted fortran sequential recs
- ? **fbindirwrite**: write specified record; fortran direct access
- ? **fbinwrite**: write a binary file containing a single record
- ? **cbinwrite**: write binary file ; mimics C block IO “write”

- **2 procedure to write ascii data**

- ? **asciwrite**: write a file containing ASCII characters
 - ✍ writes a single flat ASCII file. One value per line.
 - ✍ No user control of format
- ? **write_matrix**: write a multi-dim array to std out or to a file
 - ✍ user has format control ... pretty-print
 - ✍ options for title and row numbering

- use Fortran/C to write complicated ASCII files.

Example: binary ==> netCDF

```
begin
; read in data
  lat  = fbincread (“./in.bin”, 2, 64, “double”)
  lon  = fbincread (“./in.bin”, 3, 128, “double”)
  u    = fbincread (“./in.bin”, 6, (/64, 128/), “double”)
; assign lat/lon named dimensions and attributes
  lat!0          = “lat”
  lat@long_name = “latitude” ; lat@units = “degrees_north”
  lon!0          = “lon”
  lon@long_name = “longitude” ; lon@units = “degrees_east”
; assign t named dimensions, coordinate variables and attributes
  u!0          = “lat”
  u!1          = “lon”
  u&lat       = lat
  u&lon       = lon
  u@long_name = “zonal wind”
  u@units     = “m/s”
; create output file and output zonal wind
  fout = addfile (“out.nc”, “c”)
  fout->U = u
end
; ncl < IO_ex12.ncl
```

Example: Reading in ASCII

1881	-999.9	-999.9	-999.9	-999.9	-999.9	-999.9	999.9	-999.9
1882	-1.7	-0.5	0.6	0.1	0.9	-1.9	-3.5	-4.6
1995	-1.0	-0.8	0.4	-1.8	-1.2	-0.4	0.6	-0.1

```
begin
; read in data
  ncols   = 9
  nrows   = 3
  ksoi    = asciiread ("ascii.in", (/nrows,ncols/), "float")
; partition total array into individual vector arrays
  yrs     = ksoi(:,0)
  col1    = ksoi(:,1)
  data    = ksoi(:,1:) ; all but leftmost column
; if you were going to plot, you should assign ;named
dimensions and coordinate variables
  data@_FillValue = -999.9
end ; ncl < IO_ex15.ncl (ascii to netCDF)
```

write_matrix(x[*][*], fmt, opt)

- **pretty-print 2D array to standard out**
 - integer, float, double
 - user format control (fmt)
 - if not 2D use T=onedtond(ndtooned(TT), (/N,M/))
 - T(7,5): **write_matrix** (T, "5f7.2", False)

4.35	4.39	0.27	-3.35	-6.90
4.36	4.66	3.77	-1.66	4.06
9.73	-5.84	0.89	8.46	10.39
4.91	4.59	-3.09	7.55	4.56
.17	3.68	5.08	0.14	-5.63
-0.63	-4.12	-2.51	1.76	-1.43
-4.29	0.07	5.85	0.87	8.65

- **create an ASCII file**

```
opt      = True
opt@fout = "foo.ascii"      ; file name
write_matrix (T, "5f7.2", opt)
```

using system to access remote files

- **system** ("msread local_file /SHEA/NCEP/Test")
- **system** ("mswrite "+ fi(i) + " "+ MssPath+ fi(i))
- **system** ("msrcp")

```
begin
  mssPath = "/SHEA/NCEP/"
  fnames = (/ "reAnal1" ,"reAnal2" /)
  do nf = 0,dimsizes(fnames)-1
    system("msread "+ fnames(nf)+" "+mssPath+fnames(nf) )
    f = addfile(fnames(nf)+".grb", "r")
    .....
  end do
end
```

multiple files

- **systemfunc** (returns info from unix)
 - ? fnames = **systemfunc** ("ls reAnal*")
 - ✍ fpath = **systemfunc**("ls /mydata/reAnal*") ; full path
 - ✍ fils = **systemfunc**("cd "+path+ " ; ls reAnal*")
where: path = "/my/data/"
- **manually**
 - ? fnames = (/ "file1" , "file2" , ... /)

```
begin
  path      = "/fs/cgd/data0/shea/"
  fnames    = (/ "reAnal1", "reAnal2", "reAnal3", "reAnal4"/)
  nfiles    = dimsizes(fnames)
  do nf =0, nfiles-1
    f = addfile (path+fnames(nf)+".grb", "r")
    .....
  end do
end
```

addfiles (1 of 2)

- spans **multiple** files

- **q** = **addfiles** (**fNames**, "r")
 - ? **fNames** is a 1D array of file names (strings)
 - ? can be used for **any supported format**
 - ? technically, "q" is a variable of type **list**

T = q[:]->T

- ? read T [**values only**] from each file in list 'q'
 - ✍ T will not have any meta data
- ? T must exist in each file and be same shape [rank]
- ? a **list** is used to sequence results of addfiles
- ? normal file variable selection is used with "[...]"

addfiles (2 of 2)

- 2 options on variable merging

- ? **ListSetType** (a, "cat") [default; "cat" => concatenation]

- ? **ListSetType** (a, "join")

- when to use "cat" and "join" [rule of thumb]

- ? **cat**: continuous record

- ? **join**: creating ensembles

- ✍ a record dimension will be added

- suggest using **addfiles_GetVar** [contributed.ncl]

- ? attaches meta data to variable

Example: Read "T" across 5 files ["cat"] [Each file has 12 months]

```
begin
  fils = systemfunc ("ls "./ann*.nc")
  f     = addfiles (fils, "r")
  ListSetType(f, "cat")           ; not necessary [default]
  T     = f[:]->T
  printVarSummary(T)
end                               ; ncl < IO_ex20.ncl
```

```
----- no meta data with T
Variable: T
Type: float
Total Size: 5529600 bytes
             1382400 values
Number of Dimensions: 4
Dimensions and sizes: [60] x [5] x [48] x [96]
Coordinates:
```

Example: `addfiles_GetVar` (`contributed.ncl`)

```
load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
begin
  fils      = systemfunc ("ls ./ann*.nc")
  f         = addfiles (fils, "r")
  T         = addfiles_GetVar (f, fils, "T")
  printVarSummary (T)
end                                               ; ncl < IO_ex21.ncl
```

```
Variable: T
Type: float
Total Size: 5529600 bytes
           1382400 values
Attributes: 2
  units:      K
  long_name:  temp
Number of Dimensions: 4
Dimensions and sizes: [time|60] x [lev|5] x [lat | 48] x [lon
| 96]
Coordinates:
time: [2349 ... 4123]    lat:  [-87.159..87.159]
lev:  [85000 ... 25000] lon:  [0..356.25]
```

addfiles: option ["join"]

```
begin
  fils = systemfunc ("ls ./ann*.nc")
  f     = addfiles (fils, "r")
  ListSetType (f, "join")
  T     = f[:]->T
  printVarSummary (T)
end
```

Variable: T
Type: float
Total Size: 5529600 bytes
 1382400 values
Number of Dimensions: 5
Dimensions and sizes: [5] x [12] x [5] x [48] x [96]
Coordinates:

addfiles_GetVar (contributed.ncl)

```
load "$NCAR_ROOT/lib/ncarg/nclscripts/csm/contributed.ncl"
begin
  fils = systemfunc ("ls ./ann*.nc")
  f = addfiles (fils, "r")
  ListSetType (f, "join")
  T = addfiles_GetVar (f, fils, "T")
  printVarSummary (T)
end
```

```
Variable: T
Type: float
Total Size: 5529600 bytes
           1382400 values
Attributes: 2
  units:      K
  long_name:  temperature
Number of Dimensions: 4
Dim/sizes:  [case | 5] x [time|12] x [lev|5] x [lat | 48] x [lon | 96]
Coordinates:
time: [2349 ... 2683]    lat: [-87.159..87.159]
lev:  [85000 ... 25000] lon: [0..356.25]
```